Amendments to the Specification:

Please amend the Title as follows:

Method and System of a Microprocessor Subtraction-Division Floating Point Divider.

Please replace paragraph [0006] with the following amended paragraph:

[0006] In broad terms, performing a floating-point division operation comprises the following steps:

Step	Operation
1	$E_r = E_a - E_b$ ~~$+ E_b$~~ $+ E_{bias}$ (find the power of the quotient)
2	$S_r = S_a$ □ S_b (exclusive-OR the sign bits of each mantissa to determine the sign of the division operation)
3	$F_r = F_a \div F_b$ (In SRT, performed by iterative subtraction of multiples of *Fb* from a working partial remainder which is then shifted up. The working partial remainder initialized with F_a.)
4	Rounding of the fractional result to the most significant digit.
5	Normalize F_r (that is, $1 \leq F_r < 2$)
6	Detect underflow (if E_r is less than one) or overflow (if E_r is less than 1,023 (for double precision) or 127 (for single precision))

TABLE 1

While all these steps are required for each floating-point division operation, it is step three that is the primary concern of this specification.

Please replace paragraph [0028] with the following amended paragraph:

[0028] The number of quotient divisor multiples is dependent upon the radix of the floating-point division system. In general terms, the quotient digits fall within the set:

$$Q\varepsilon \{-(R-1), -(R-2), \ldots, -(R-N), -0, 0\ (R-N), \ldots, (R-2), (R-1), R\}$$

where Q is the set of possible quotient digits, R is the radix of the floating-point division system, and $N=R-1$. Thus, the prescale unit 60 preferably computes, in the general form, $(R-1)D$ (positive and negative), $(R-2)D$ (positive and negative) through *RD*. Consider the preferred Radix-4 implementation where the quotient digit set is:

$Q\varepsilon \{-3,-2,-1,-0,0,1,2,3,4\}$

In this case, the prescaling unit preferably calculates *3D* (both positive and negative), *2D* (both positive and negative), and *4D*. The zero multiple is straight-forward, and the negative zero multiple is merely the two's ~~compliment~~ complement version of the zero multiple. Moreover, the single multiple of *D* is a pass-through operation. Both the *2D* and *4D* multiples are shift operations, and thus, in the radix-4 system, only the *3D* multiple requires an actual multiplication. Thus, quotient digits fall within the set, and the prescale unit 60 preferably calculates a divisor multiple based upon each quotient digit in the set prior to the first SD cell performing its operation.

Please replace paragraph [0032] with the following amended paragraph:

[0032] In order to speed the computation time through each SD stage or cell, quotient divisor multiples are preferably pre-calculated in the prescaling unit 60 and are made available to each SD stage prior to or at least contemporaneous with that SD stage performing its calculation. Referring again to Figure 2, the adders 100 are coupled to the multiplexers 102. Each multiplexer 102 has as its inputs respective bits of each of the quotient divisor multiples. Based on the value of the select lines for each of the multiplexers 102, one input couples through the multiplexer and thus to the adders 100. A quotient digit selected by the previous SD stage drives the select lines of the multiplexers 102, thus coupling the appropriate quotient divisor multiple to the adders 100. In the case of a first SD stage 62, the first quotient digit is determined by the scaled dividend. Thus, each of the adders 100 has as one input respective bits of the selected divisor multiple, and has as a second input respective bits of the partial remainder from the previous stage (some of the bits of which may be represented in carry-save form, which is discussed more fully below). The adders 100 perform the addition to obtain a resultant in carry-save form. One of ordinary skill in the art understands that the subtraction required to determine the next partial remainder may be accomplished with the adders 100 by adding the two's ~~compliment~~ complement version of the divisor multiples.

Please replace paragraph [0036] with the following amended paragraph:

[0036] The two most significant bits to the right of the radix point 104 of the resultant, along with their carry, indicate the quotient digit for the next SD stage. That is, the combination of R^{-1} and R^{-2}, or their ~~compliment~~complement, indicate the value of the next quotient digit directly with C_0 124 and C_1 122 determining the sign. In the special case where both C_0 124 and C_1 122 are not asserted, these digits become the controlling factors in the quotient digit selection. Table 4 below defines a truth table for possible values of C_0, C_1, R^{-1}, R^{-2} and the next quotient digit Q_{i+1} for a radix for a Radix-4 implementation.

C_0	C_1	R^{-1}	R^{-2}	C_0 AND C_1	C_0 XOR C_1	Q_{i+1} decimal
0	0	0	0	0	0	-3
0	0	0	1	0	0	-2
0	0	1	0	0	0	-1
0	0	1	1	0	0	-0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	2
0	1	1	1	0	1	3
1	0	0	0	0	1	0
1	0	0	1	0	1	1
1	0	1	0	0	1	2
1	0	1	1	0	1	3
1	1	0	0	1	0	4
1	1	0	1	1	0	d/c
1	1	1	0	1	0	d/c
1	1	1	1	1	0	d/c

TABLE 4

As shown in Table 4, if (C_0 AND C_1) is not asserted, and (C_0 XOR C_1) is asserted, the next quotient digit is the value of the combination of R^{-1} and R^{-2} (shown in the table in the Q_{i+1} table in decimal form). If (C_0 AND C_1) is not asserted, and (C_0 XOR C_1) is not asserted, the next quotient digit is the negative value of the one's ~~compliment~~ complement of R^{-1} and R^{-2}. Finally, when (C_0 AND C_1) is asserted, the next quotient digit is the value of the radix, in the Radix-4 the next quotient digit is four. The final three entries in Table 4 are "don't care" (d/c) conditions as the R^{-1} and R^{-2} are zero in the preferred embodiments when (C_0 AND C_1) is asserted. Thus, the decode logic 126 takes into account the value of C_0 124, C_1 122, R^{-1} 120 and R^{-2} 118 to produce a multiplexer select output for the next SRT stage. As indicated by Figure 2, preferably the multiplexer select for the next SD stage has a plurality of signals coupled to the select lines of each multiplexer in the next stage, and indeed it is preferred that these select lines are fully decoded.

Please replace paragraph [0041] with the following amended paragraph:

[0041] In broad terms, the decode logic 238 performs addition of two incoming three bit numbers, and produces an output in ~~one-hot-hot~~one-hot encoded form. Again, implementing addition as a decode 0-14 logic 238 takes more hardware than full adder counterparts, but the fully decoded output 242 forms the basis for shift operations within carry-detect logic 240 (to be discussed more fully below). Preferably, decode logic 238 begins the process of calculating the estimated partial remainders for the particular SD cell in advance. Indeed, in the preferred embodiment, decode logic 238 operates substantially simultaneously with the equivalent of adder logic 206 of the previous SD cell. Thus, preferably the fully decoded outputs 242 of decode logic 238 are available before the adders 200 have had an opportunity to perform their add operation.

Please replace paragraph [0045] with the following amended paragraph:

[0045] Operation of the remainder of the implementation shown in Figure 3 is substantially the same as that described with respect to Figure 2. In particular, the adders 200 add the partial remainder from the previous SD stage or cell to the two's ~~compliment~~ complement version of the divisor multiples to obtain, in carry-save form, the resultant for the present SD cell. The R^{-3} and R^{-4} carry-save representations are preferably converted in logic 206 to their carry-propagate form so as to feed the decode 0-6 logic of the next SD stage.